P&S Modern SSDs

FLIN:

Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives

Arash Tavakkol, Mohammad Sadrosadati, Saugata Ghose,
Jeremie S. Kim, Yixin Luo, Yaohua Wang, Nika Mansouri Ghiasi,
Lois Orosa, Juan Gómez-Luna, Onur Mutlu

ISCA 2018

Executive Summary

- Modern solid-state drives (SSDs) use new storage protocols (e.g., NVMe) that **eliminate the OS software stack**
  - I/O requests are now scheduled inside the SSD
  - Enables **high throughput:** millions of IOPS
- OS software stack elimination **removes existing fairness mechanisms**
  - We **experimentally characterize fairness** on four real state-of-the-art SSDs
  - **Highly unfair slowdowns:** large difference across concurrently-running applications
- We find and analyze **four sources of inter-application interference** that lead to slowdowns in state-of-the-art SSDs
- **FLIN:** a new I/O request scheduler for modern SSDs designed to **provide both fairness and high performance**
  - Mitigates all four sources of inter-application interference
  - Implemented fully in the SSD controller firmware, uses < 0.06% of DRAM space
  - FLIN improves **fairness by 70%** and **performance** by **47%** compared to a state-of-the-art I/O scheduler

Background: Modern SSD Design

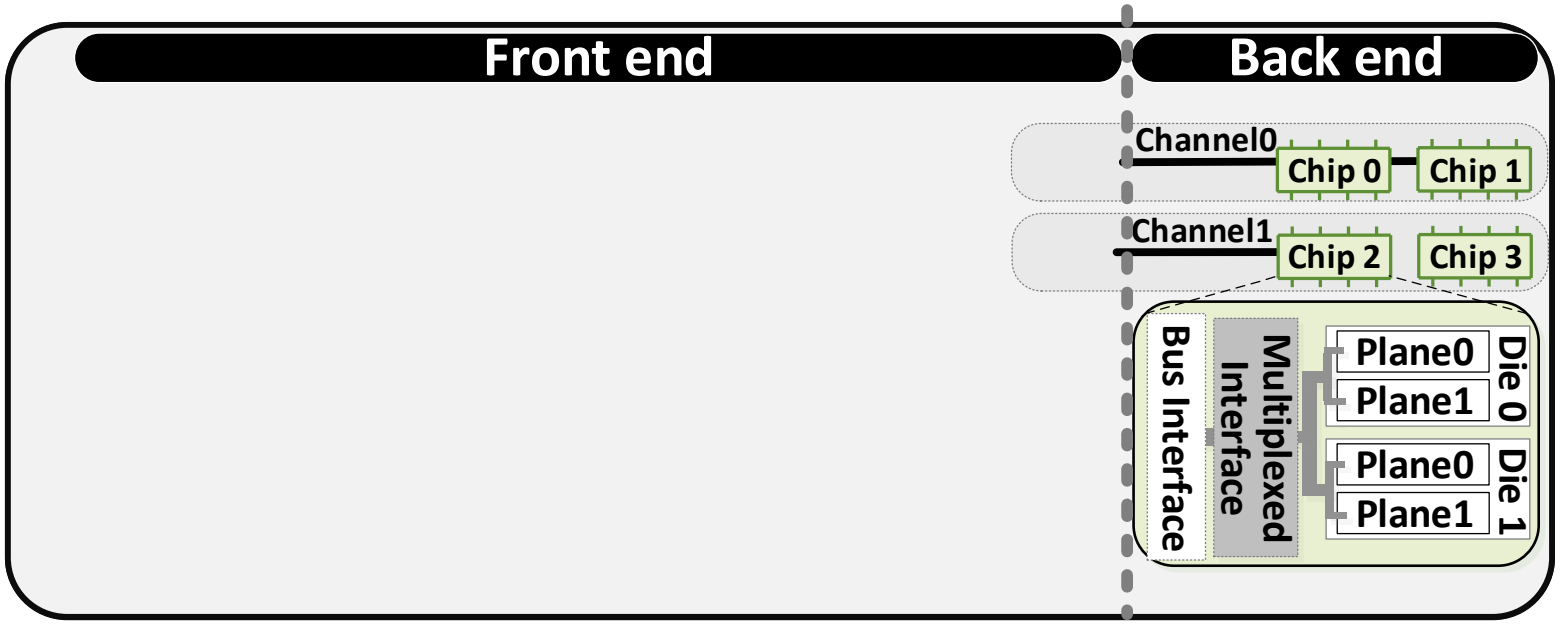
Unfairness Across Multiple Applications in Modern SSDs

FLIN:
Flash-Level INterference-aware SSD Scheduler

Experimental Evaluation

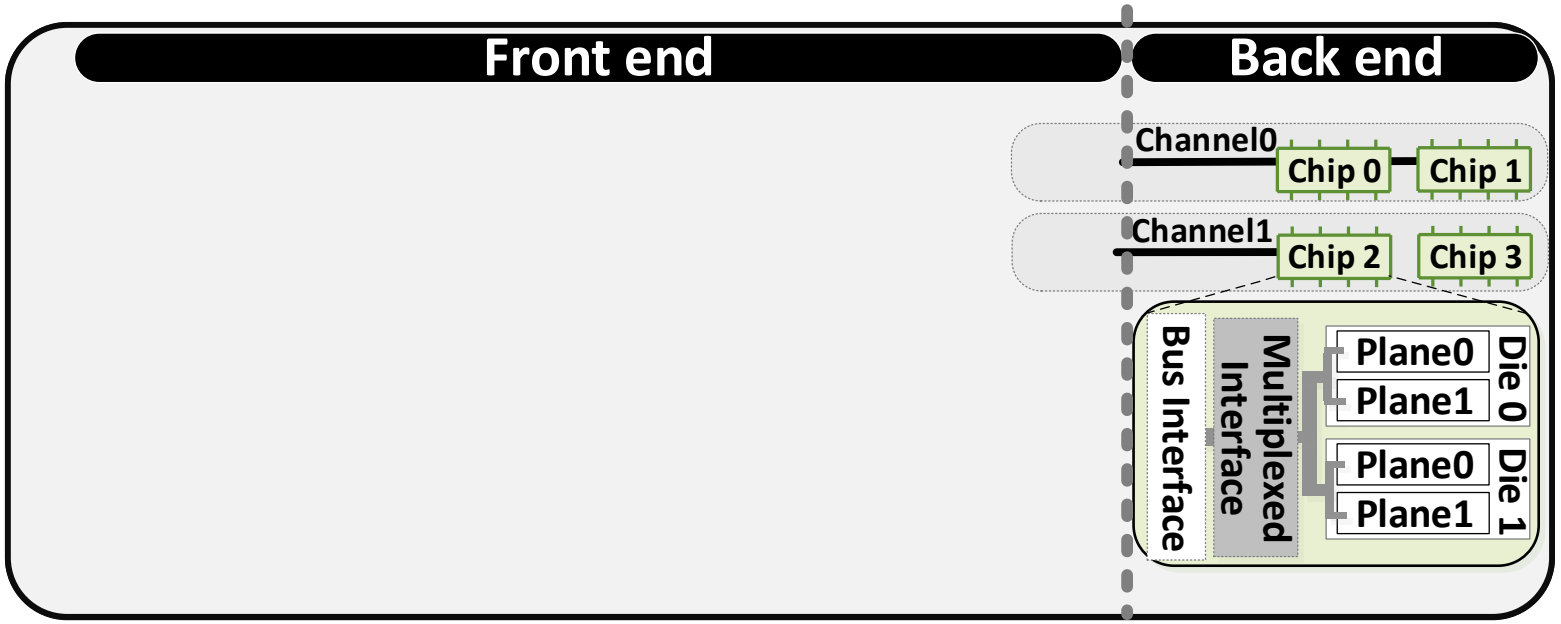
Conclusion

Internal Components of a Modern SSD



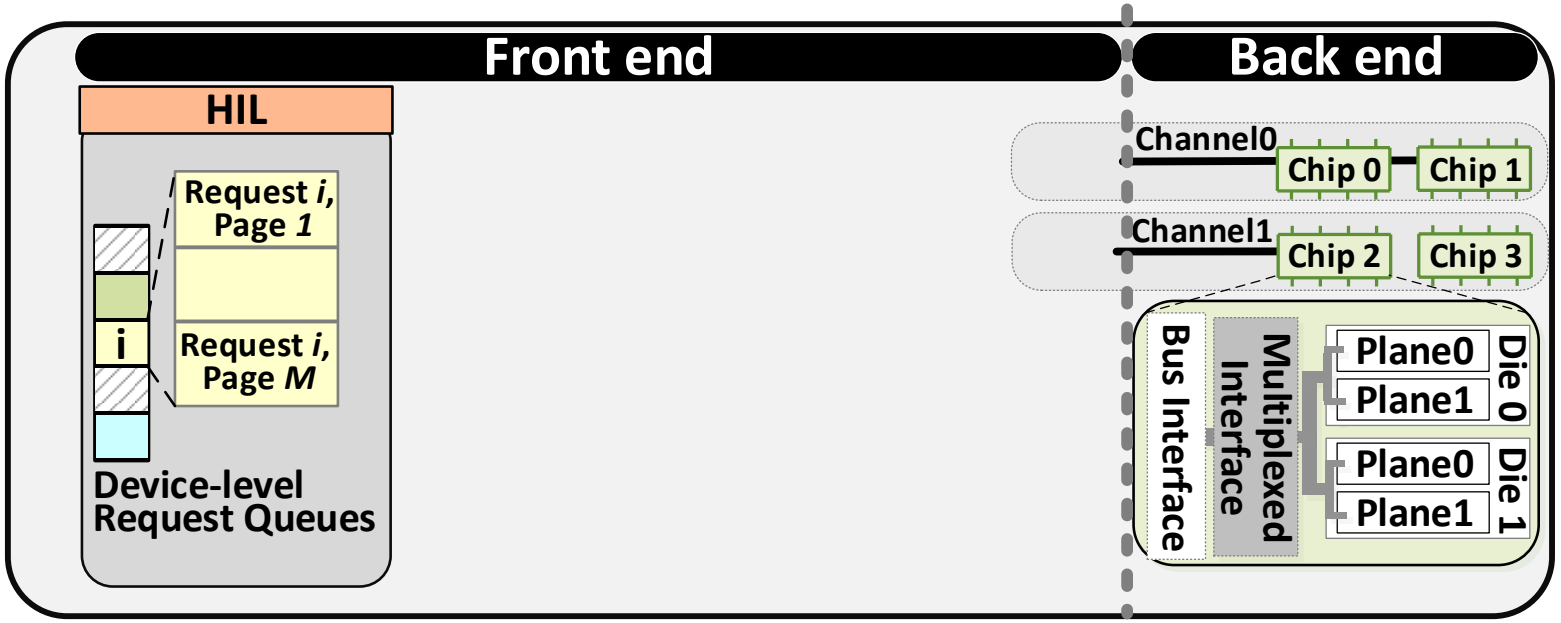
- **Back End:** data storage
  - Memory chips (e.g., NAND flash memory, PCM, MRAM, 3D XPoint)

Internal Components of a Modern SSD



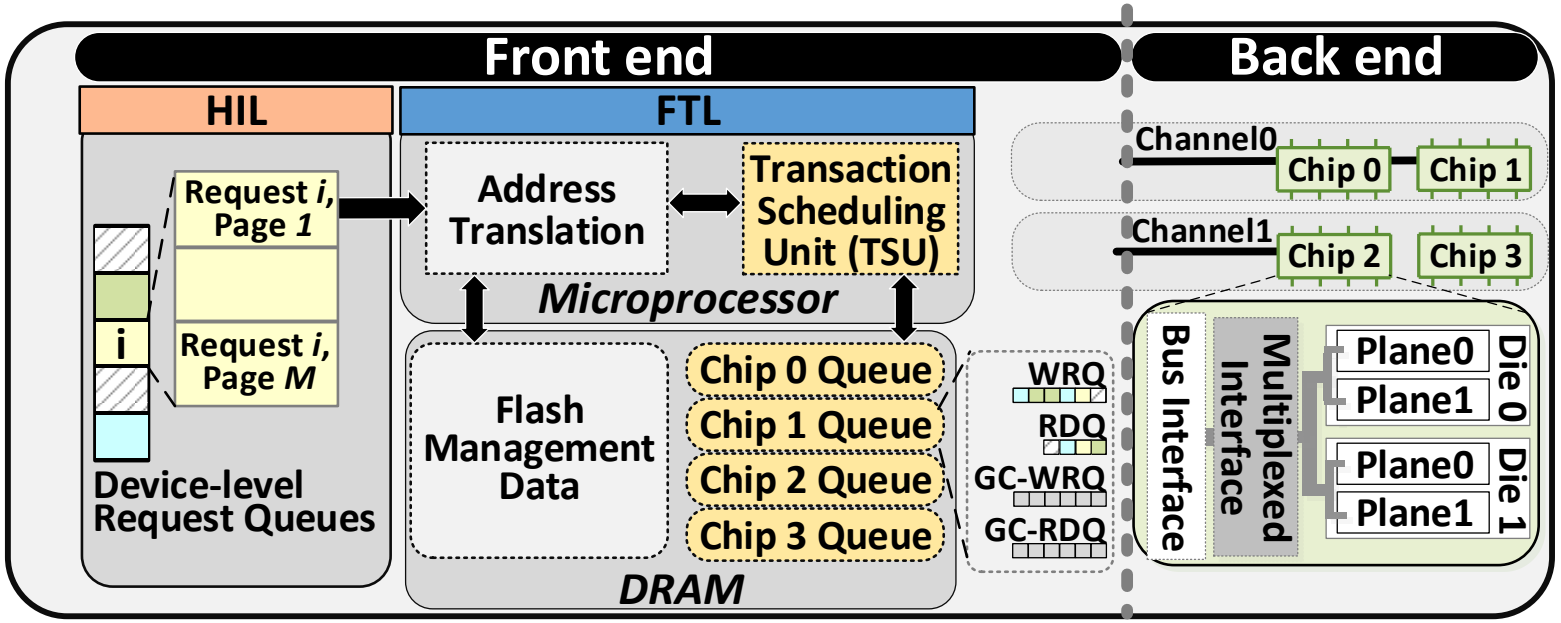
- **Back End:** data storage
  - Memory chips (e.g., NAND flash memory, PCM, MRAM, 3D XPoint)
- **Front End:** management and control units

Internal Components of a Modern SSD



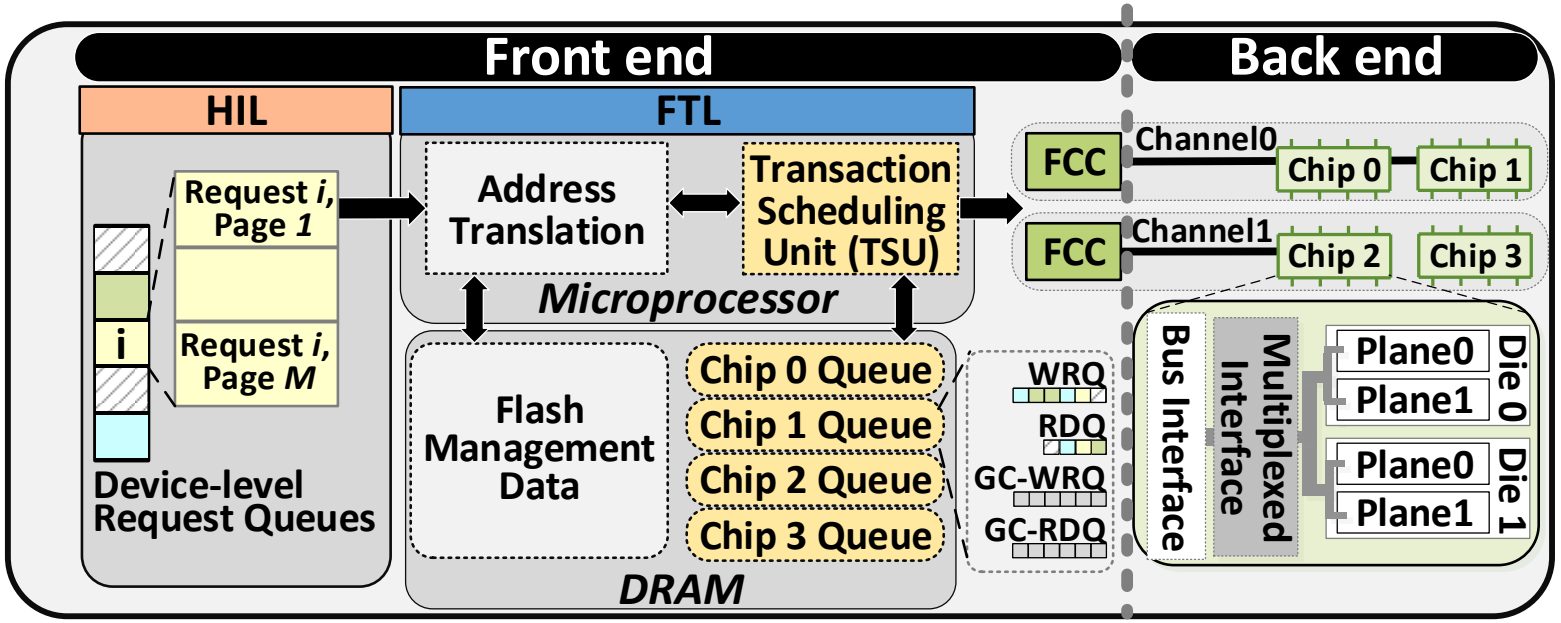
- **Back End:** data storage
  - Memory chips (e.g., NAND flash memory, PCM, MRAM, 3D XPoint)
- **Front End:** management and control units
  - **Host–Interface Logic (HIL):** protocol used to communicate with host

Internal Components of a Modern SSD



- **Back End:** data storage
  - Memory chips (e.g., NAND flash memory, PCM, MRAM, 3D XPoint)
- **Front End:** management and control units
  - Host–Interface Logic (HIL): protocol used to communicate with host
  - **Flash Translation Layer (FTL):** manages resources, processes I/O requests

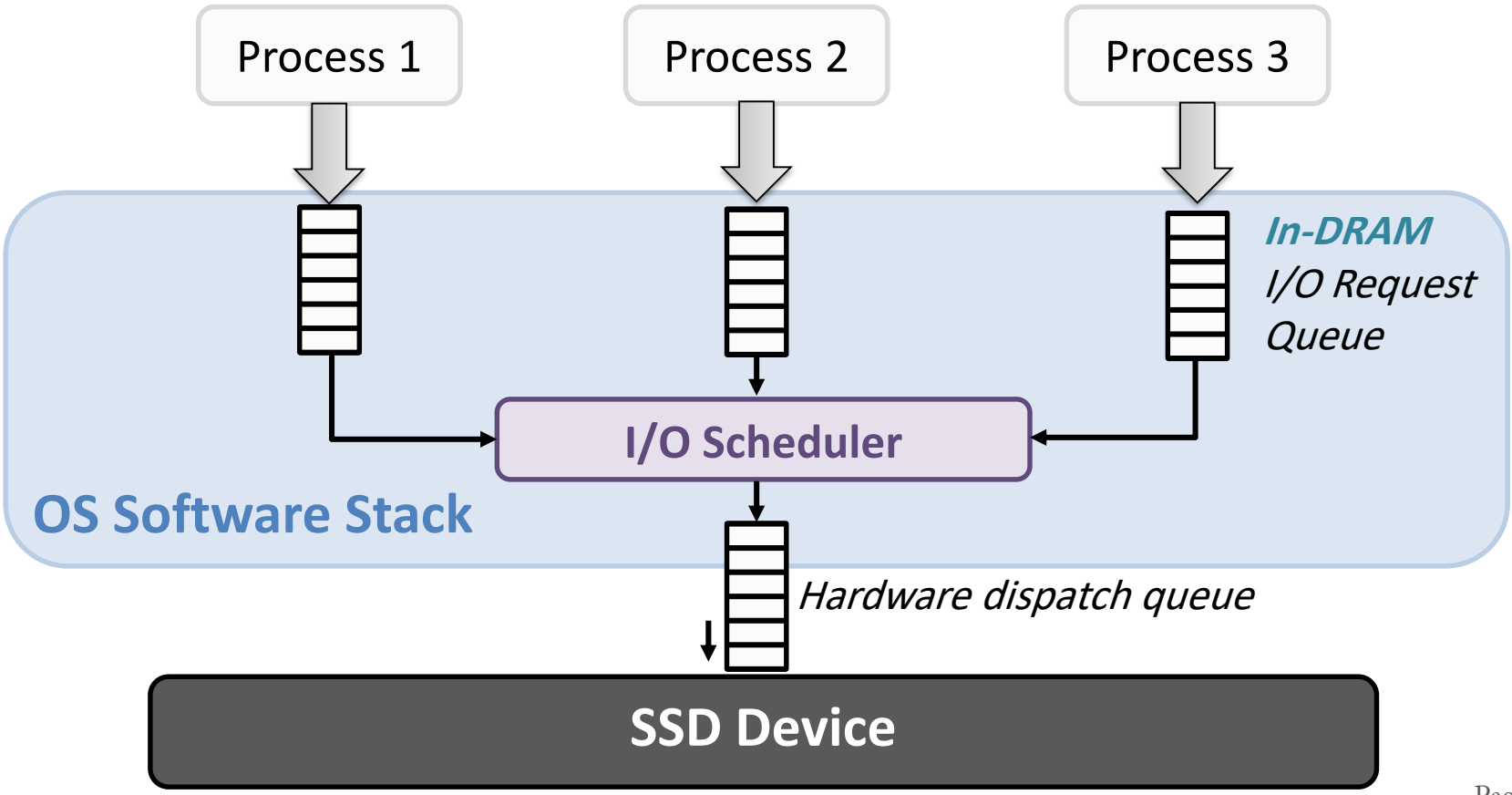
Internal Components of a Modern SSD



- **Back End:** data storage
  - Memory chips (e.g., NAND flash memory, PCM, MRAM, 3D XPoint)
- **Front End:** management and control units
  - Host–Interface Logic (HIL): protocol used to communicate with host
  - Flash Translation Layer (FTL): manages resources, processes I/O requests
  - **Flash Channel Controllers (FCCs):** sends commands to, transfers data with memory chips in back end

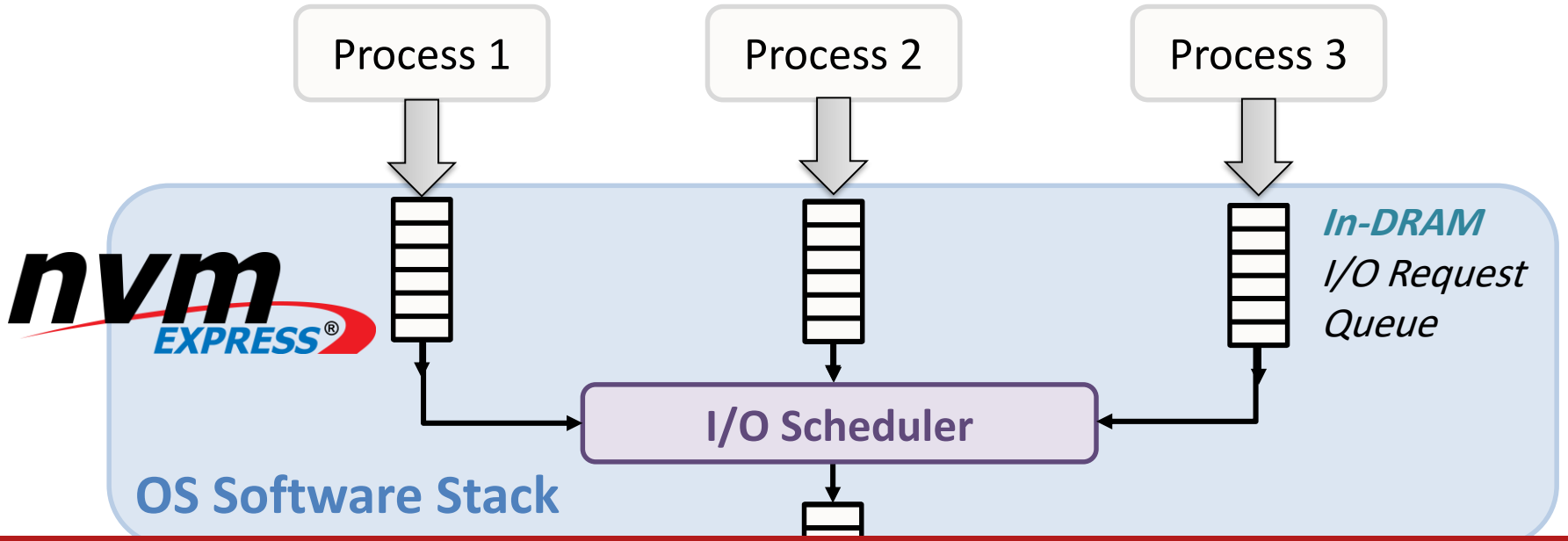
Conventional Host–Interface Protocols for SSDs

- SSDs initially adopted **conventional** host–interface protocols (e.g., SATA)
  - Designed for magnetic hard disk drives
  - Maximum of only ***thousands*** **of IOPS** per device



Host–Interface Protocols in Modern SSDs

- Modern SSDs use high-performance host–interface protocols (e.g., NVMe)
  - Bypass OS intervention: SSD must perform scheduling
  - Take advantage of SSD throughput: enables *millions* of IOPS per device



Fairness mechanisms in OS software stack are also eliminated

Do modern SSDs need to handle fairness control?

Background: Modern SSD Design

Unfairness Across Multiple Applications in Modern SSDs

FLIN:
Flash-Level INterference-aware SSD Scheduler

Experimental Evaluation

Conclusion

Measuring Unfairness in Real, Modern SSDs

- We measure fairness using four real state-of-the-art SSDs
  - NVMe protocol
  - Designed for datacenters

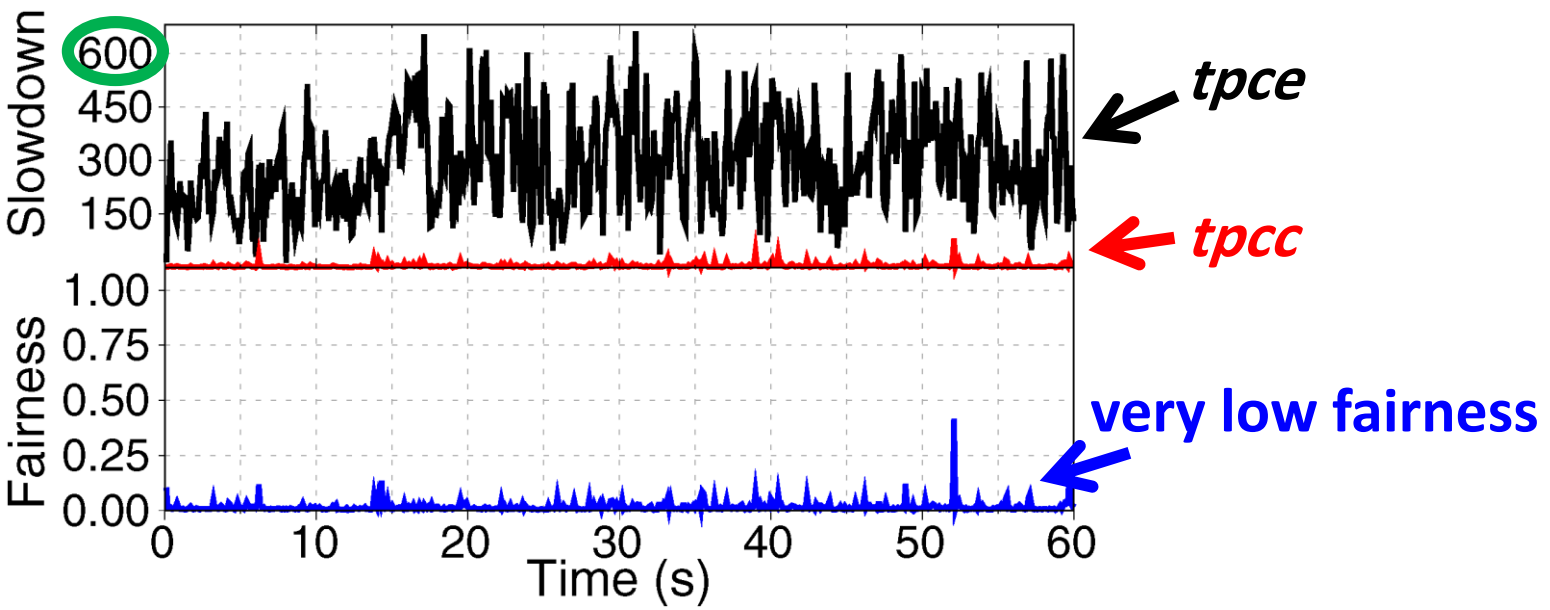
- Flow: a series of I/O requests generated by an application

- Slowdown $= \frac{\text{shared flow response time}}{\text{alone flow response time}}$ *(lower is better)*

- Unfairness $= \frac{\text{max slowdown}}{\text{min slowdown}}$ *(lower is better)*

- Fairness $= \frac{1}{\text{unfairness}}$ *(higher is better)*

Representative Example: *tpcc* and *tpce*



average slowdown of *tpce*:
2x to 106x across our four real SSDs

SSDs do not provide fairness among concurrently-running flows

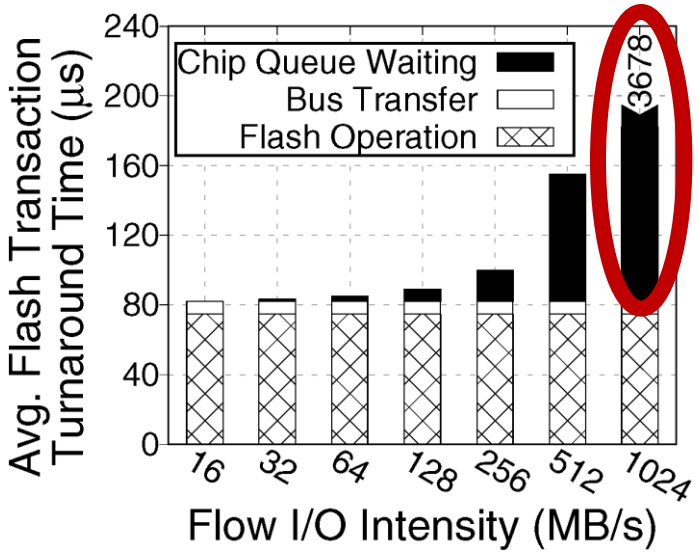
What Causes This Unfairness?

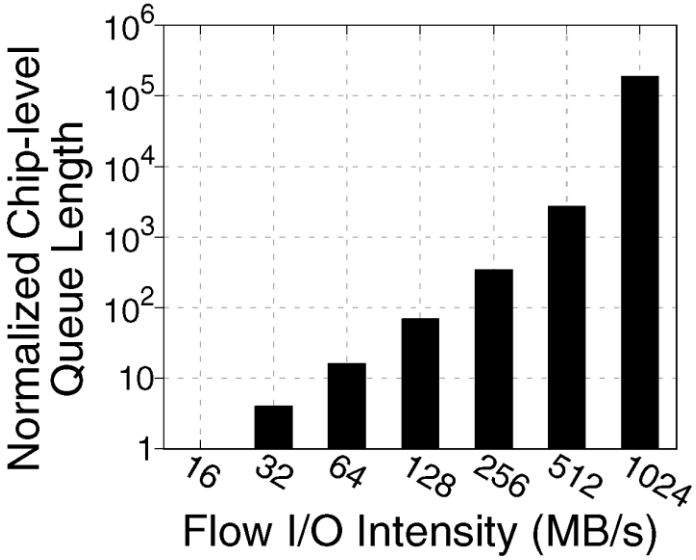
- Interference among concurrently-running flows
- We perform a detailed study of interference
  - MQSim: detailed, open-source modern SSD simulator [FAST 2018] https://github.com/CMU-SAFARI/MQSim
  - Run flows that are designed to demonstrate each source of interference
  - Detailed experimental characterization results in the paper
- We uncover four sources of interference among flows

Source 1: Different I/O Intensities

SAFARI

- The I/O intensity of a flow affects the average queue wait time of flash transactions

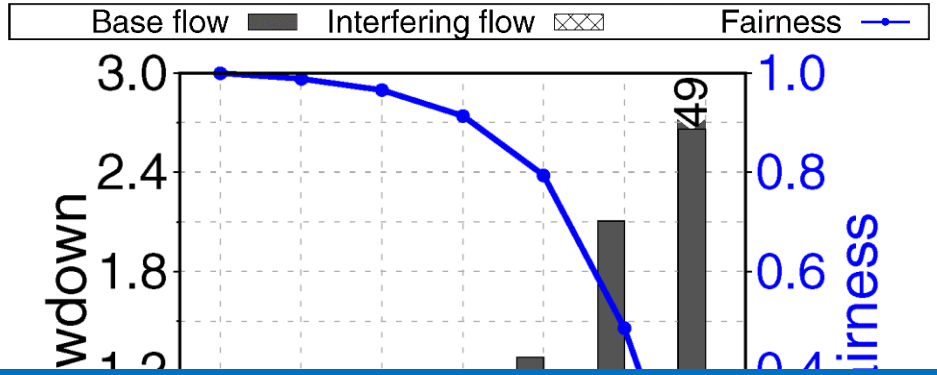




The queue wait time highly increases with I/O intensity

Source 1: Different I/O Intensities

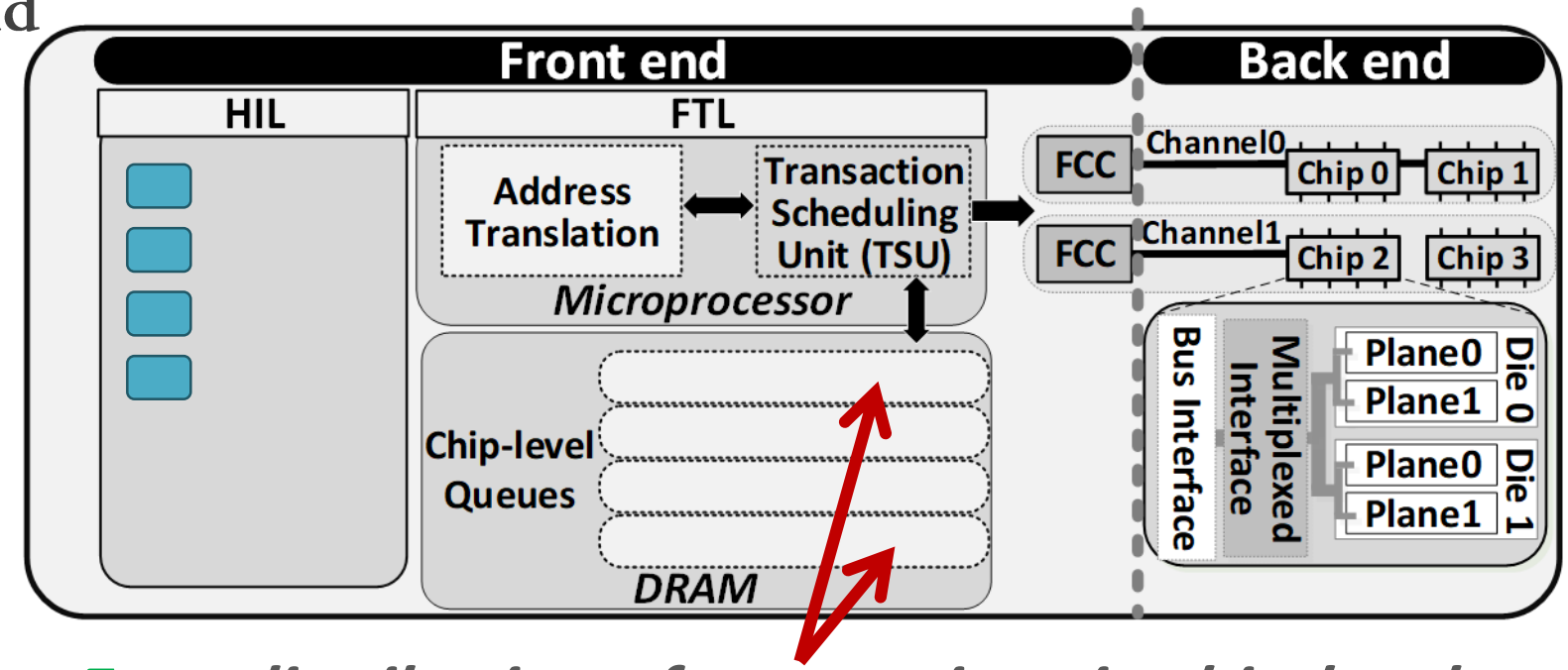
- An experiment to analyze the effect of concurrently executing two flows with **different I/O intensities** on fairness
  - **Base flow**: low intensity (16 MB/s) and low average chip-level queue length
  - **Interfering flow**: varying I/O intensities from low to very high



The average response time of a low-intensity flow substantially increases due to interference from a high-intensity flow

Source 2: Different Access Patterns

- Some flows take advantage of **chip-level parallelism** in back end



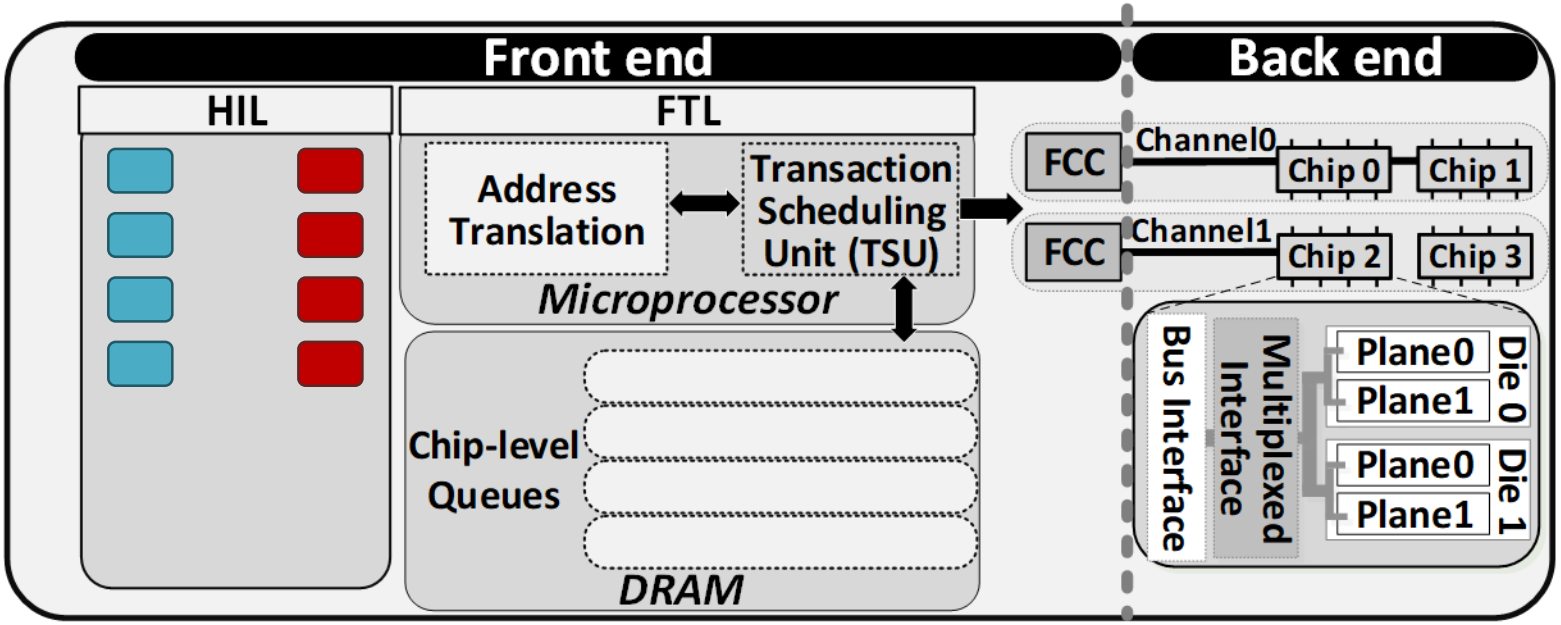
***Even** distribution of transactions in chip-level queues*

- Leads to a **low queue wait time**

Source 2: Different Request Access Patterns

SAFARI

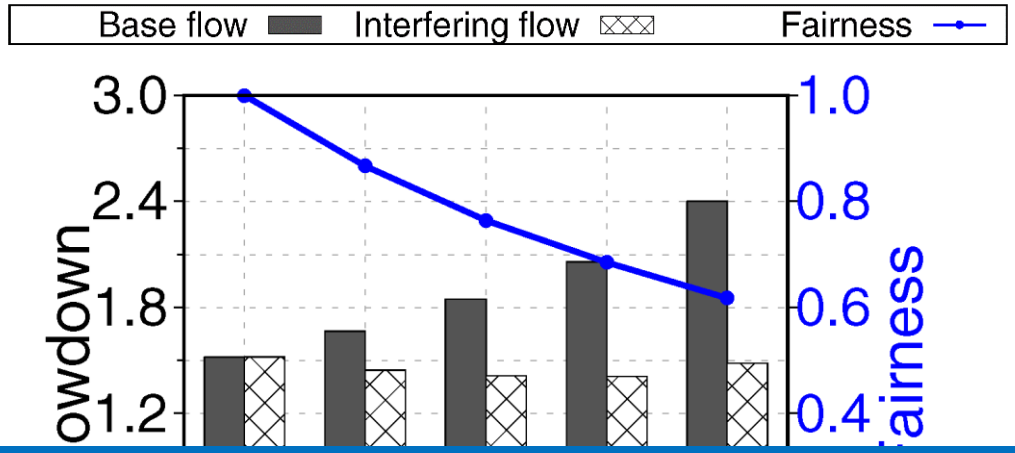
- Other flows have access patterns that **do not exploit parallelism**



Source 2: Different Request Access Patterns

SAFARI

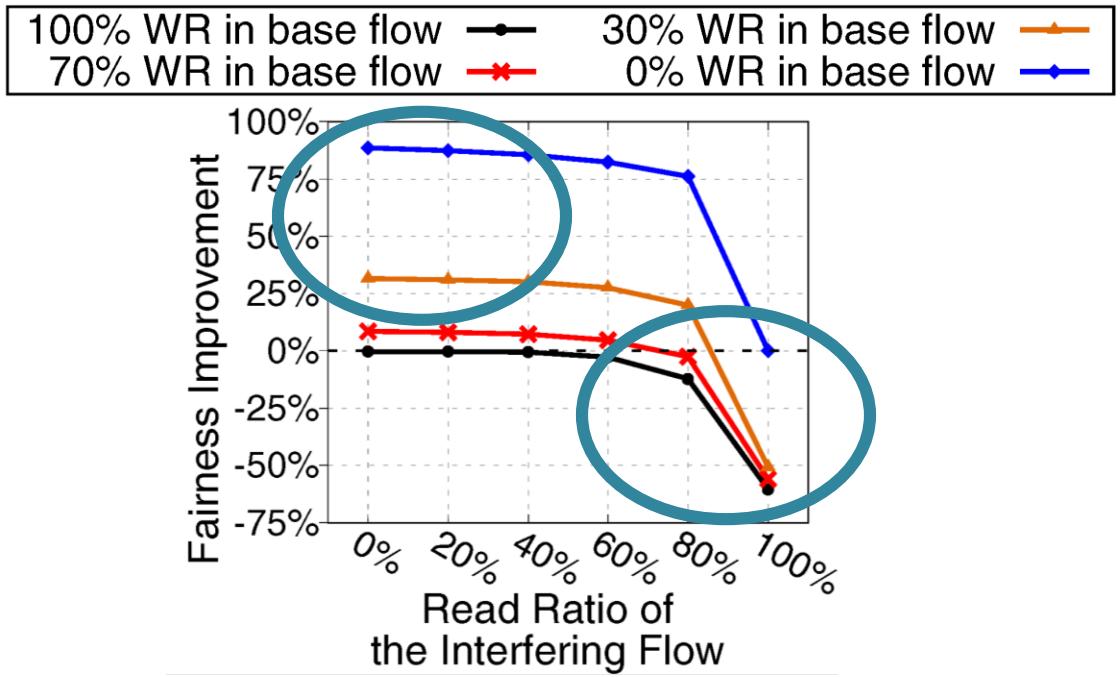
- An experiment to analyze the interference between concurrent flows with different access patterns
  - Base flow: streaming access pattern (parallelism friendly)
  - Interfering flow: mixed streaming and random access pattern



Flows with parallelism-friendly access patterns are susceptible to interference from flows whose access patterns do not exploit parallelism

Source 3: Different Read/Write Ratios

- State-of-the-art SSD I/O schedulers **prioritize reads over writes**



When flows have different read/write ratios, existing schedulers do not effectively provide fairness

Source 4: Different Garbage Collection Demands

SAFARI

- NAND flash memory performs **writes out of place**
  - Erases can only happen on an entire **flash block** (hundreds of flash pages)
  - Pages marked invalid during write
- **Garbage collection** (GC)
  - Selects a block with mostly-invalid pages
  - Moves any remaining valid pages
  - Erases blocks with mostly-invalid pages
- **High-GC flow: flows with a higher write intensity induce more garbage collection activities**

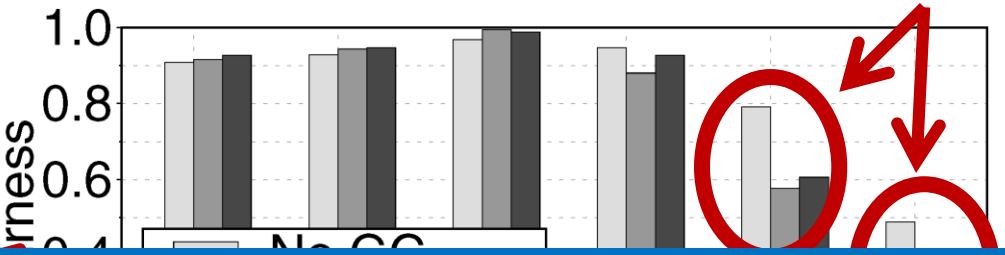
Source 4: Different Garbage Collection Demands

SAFARI

- Garbage collection may block user I/O requests
  - Primarily depends on the write intensity of the workload
- An experiment with two 100%-write flows with different intensities
  - Base flow: low intensity and moderate GC demand
  - Interfering flow: different write intensities from low-GC to high-GC

Lower fairness due to GC execution

Tries to preempt GC



The GC activities of a high-GC flow can unfairly block flash transactions of a low-GC flow

Summary: Source of Unfairness in SSDs

- Four major sources of unfairness in modern SSDs
  1. I/O intensity
  2. Request access patterns
  3. Read/write ratio
  4. Garbage collection demands

OUR GOAL

Design an I/O request scheduler for SSDs that (1) provides fairness among flows by mitigating all four sources of interference, and (2) maximizes performance and throughput

Background: Modern SSD Design

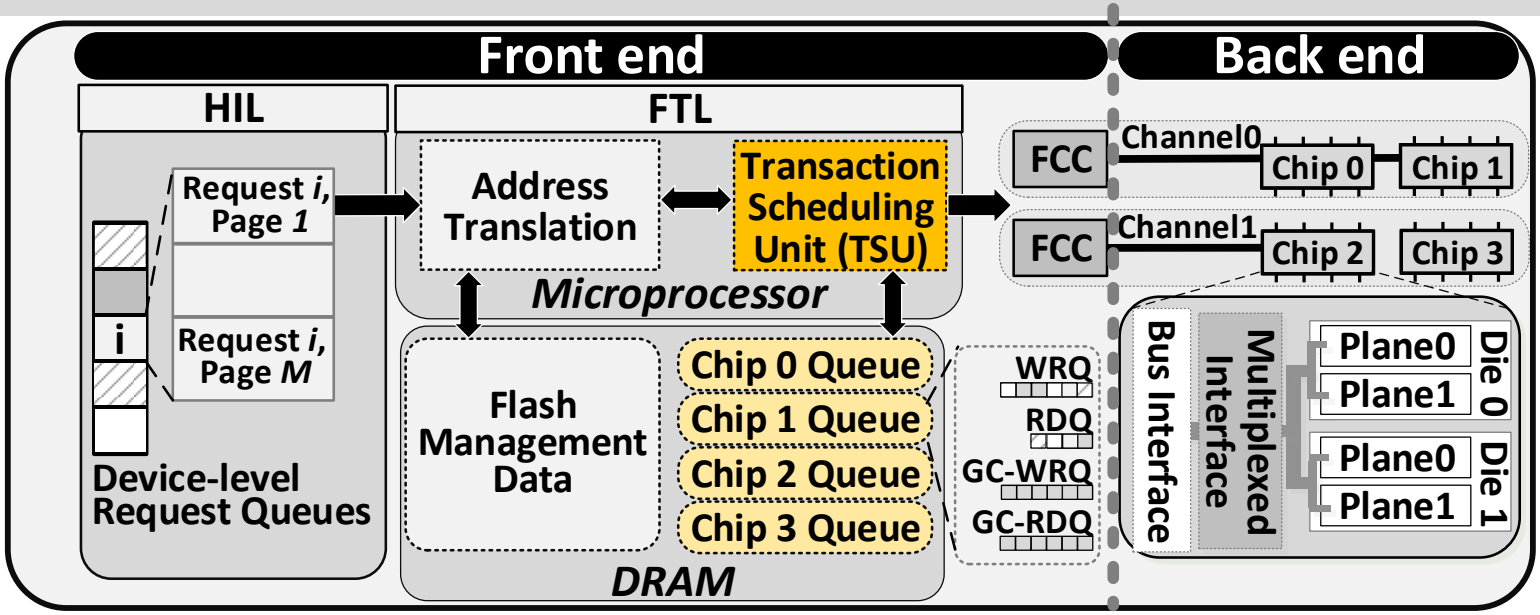
Unfairness Across Multiple Applications in Modern SSDs

FLIN:
Flash-Level INterference-aware SSD Scheduler

Experimental Evaluation

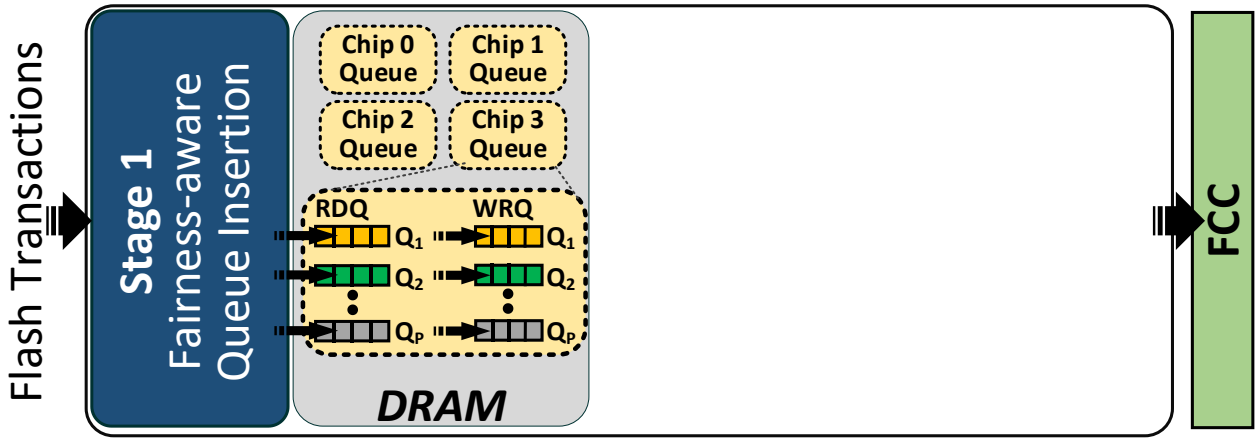
Conclusion

FLIN: Flash-Level INterference-aware Scheduler



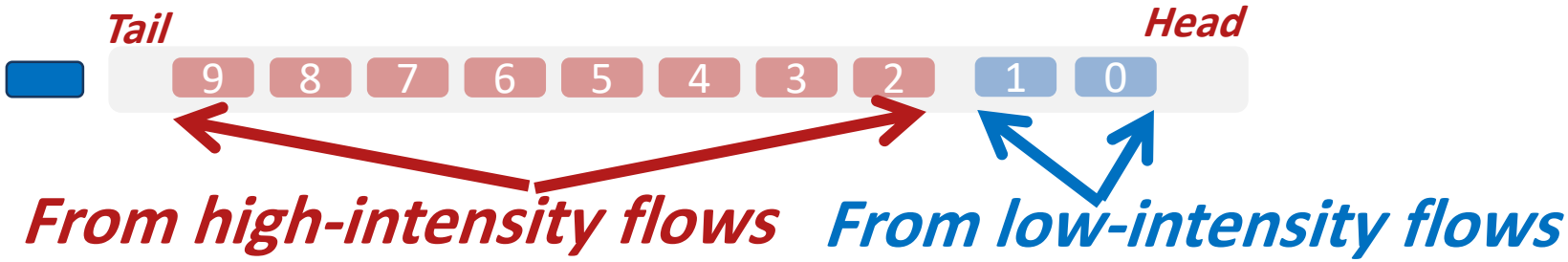
- FLIN is a three-stage I/O request scheduler
  - Replaces existing transaction scheduling unit
  - Takes in flash transactions, reorders them, sends them to flash channel
- Identical throughput to state-of-the-art schedulers
- Fully implemented in the SSD controller firmware
  - No hardware modifications
  - Requires < 0.06% of the DRAM available within the SSD

Three Stages of FLIN

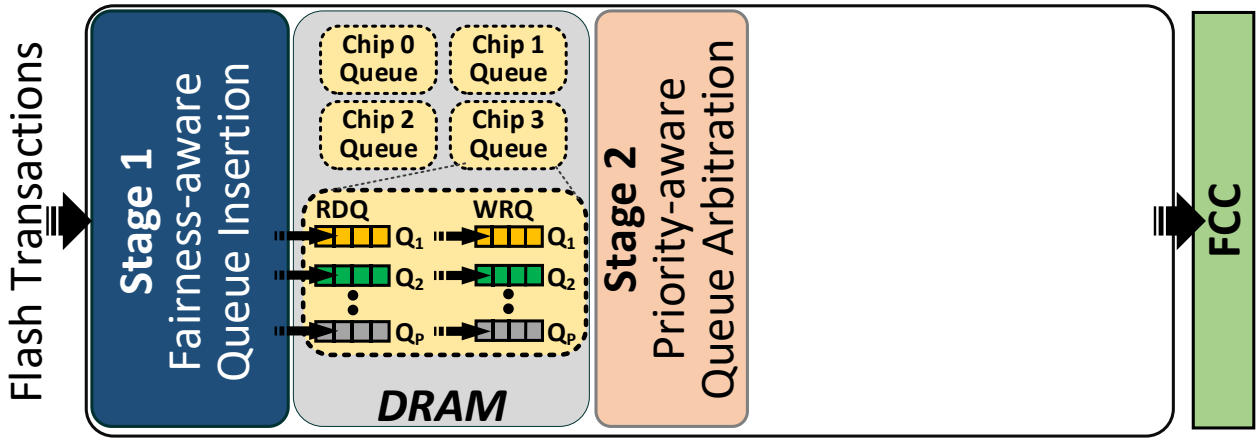


- Stage 1: Fairness-aware Queue Insertion

  relieves I/O intensity and access pattern interference



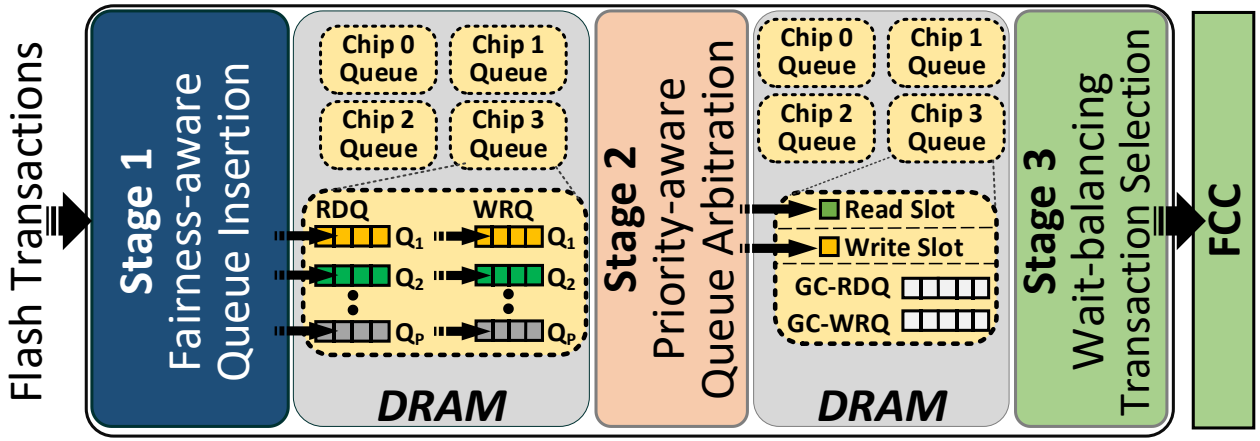
Three Stages of FLIN



- Stage 1: Fairness-aware Queue Insertion
  relieves I/O intensity and access pattern interference

- **Stage 2: Priority-aware Queue Arbitration**
  enforces priority levels that are assigned to each flow by the host

Three Stages of FLIN



- Stage 1: Fairness-aware Queue Insertion
  relieves I/O intensity and access pattern interference

- Stage 2: Priority-aware Queue Arbitration
  enforces priority levels that are assigned to each flow by the host

- Stage 3: Wait-balancing Transaction Selection
  relieves read/write ratio and garbage collection demand interference

Background: Modern SSD Design

Unfairness Across Multiple Applications in Modern SSDs

FLIN:
Flash-Level INterference-aware SSD Scheduler

Experimental Evaluation

Conclusion

Evaluation Methodology

- **MQSim: https://github.com/CMU-SAFARI/MQSim** [FAST 2018]
  - Protocol: NVMe 1.2 over PCIe
  - User capacity: 480GB
  - Organization: 8 channels, 2 planes per die, 4096 blocks per plane, 256 pages per block, 8kB page size

- **40 workloads containing four randomly-selected storage traces**
  - Each storage trace is collected from real enterprise/datacenter applications: UMass, Microsoft production/enterprise
  - Each application classified as low-interference or high-interference

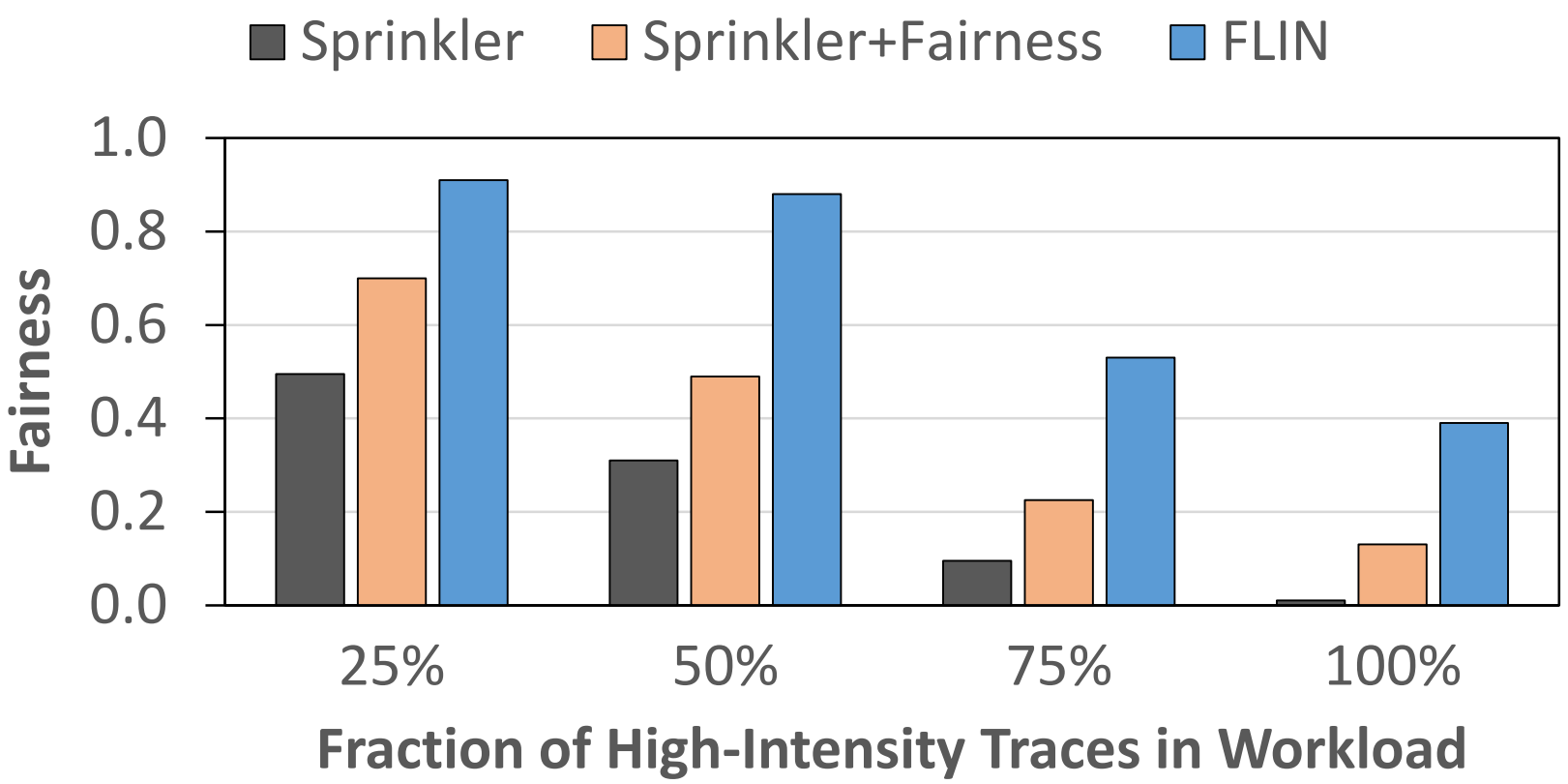
Two Baseline Schedulers

- **Sprinkler** [Jung+ HPCA 2014]
  a state-of-the-art device-level high-performance scheduler

- **Sprinkler+Fairness** [Jung+ HPCA 2014, Jun+ NVMSA 2015]
  we add a state-of-the-art fairness mechanism to Sprinkler that was previously proposed for OS-level I/O scheduling
  - **Does not have direct information** about the internal resources and mechanisms of the SSD
  - Does not mitigate **all four sources of interference**

FLIN Improves Fairness Over the Baselines

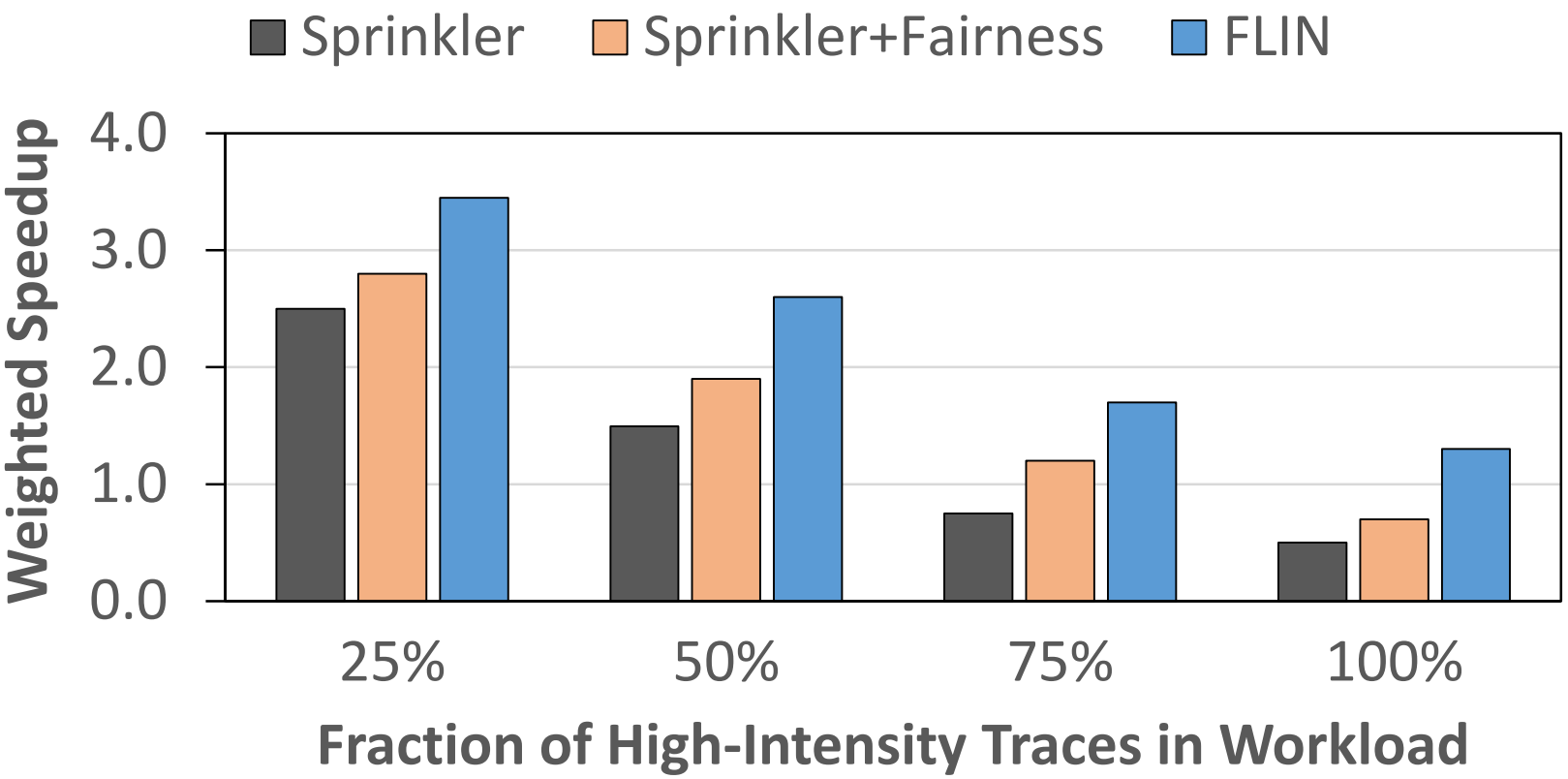
SAFARI



FLIN improves fairness by an average of 70%, by mitigating *all* four major sources of interference

FLIN Improves Performance Over the Baselines

SAFARI



FLIN improves performance by an average of 47%, by making use of idle resources in the SSD and improving the performance of low-interference flows

Other Results in the Paper

- **Fairness and weighted speedup for each workload**
  - FLIN improves fairness and performance for *all* workloads
- **Maximum slowdown**
  - Sprinkler/Sprinkler+Fairness: several applications with maximum slowdown over 500x
  - FLIN: no flow with a maximum slowdown over 80x
- **Effect of each stage of FLIN on fairness and performance**
- **Sensitivity study to FLIN and SSD parameters**
- **Effect of write caching**

Background: Modern SSD Design

Unfairness Across Multiple Applications in Modern SSDs

FLIN:
Flash-Level INterference-aware SSD Scheduler

Experimental Evaluation

Conclusion

Conclusion

- Modern solid-state drives (SSDs) use new storage protocols (e.g., NVMe) that **eliminate the OS software stack**
  - Enables **high throughput:** millions of IOPS
  - OS software stack elimination **removes existing fairness mechanisms**
  - **Highly unfair slowdowns** on real state-of-the-art SSDs
- **FLIN:** a new I/O request scheduler for modern SSDs designed to **provide both fairness and high performance**
  - Mitigates all four sources of inter-application interference
    - » Different I/O intensities
    - » Different request access patterns
    - » Different read/write ratios
    - » Different garbage collection demands
  - Implemented fully in the SSD controller firmware, uses < 0.06% of DRAM
  - FLIN improves **fairness by 70%** and **performance** by **47%** compared to a state-of-the-art I/O scheduler (Sprinkler+Fairness)

P&S Modern SSDs

FLIN:

Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives

Arash Tavakkol, Mohammad Sadrosadati, Saugata Ghose,
Jeremie S. Kim, Yixin Luo, Yaohua Wang, Nika Mansouri Ghiasi,
Lois Orosa, Juan Gómez-Luna, Onur Mutlu

ISCA 2018